Compiling each said piece of said image of guest OS into multiple code segments;

Packaging each said piece of said image of guest OS into a native application of said host OS;

~~The method of claim 16, further comprising a step of splitting guest kernel image and other related files into different pieces and wrapped into multiple host OS files.~~

Claims 20-35 are withdrawn.

Remarks

This amendment is in response to the Office Action mailed October 17, 2007. The many helpful suggestions in the Office Action are highly appreciated have thereafter been incorporated in the amended claims.

Substance of Interview

Examiner requested an election via telephone and applicant elects claims 1-19.

Election/Restrictions

Examiner requires restriction to one of the following inventions is required under 35 U.S.C. 121:

I. Claims 1-19, drawn to methods for allowing a mobile device to run multiple operating systems by packaging a guest OS inside a host OS image, classified in class 713, subclass 1.

II. Claims 20-22, drawn to a method to preserve state and data of a host OS by protecting memory, classified in class 711, subclass 152.

III. Claims 23-27, drawn to a method to preserve state and data of a host OS by "faking" a reduced memory area, classified in class 711, subclass 152.

IV. Claims 28-31, drawn to a memory device driver to claim memory for a host OS, classified in class 711, subclass 147.

V. Claim 32-35, drawn to a method to preserve state and data of a host OS by backing up to an external memory card, classified in class 711, subclass 162.

During telephone interview with examiner, a provisional election was made without traverse to prosecute the invention of group I, claims 1-19.

Claims 20-35 are withdrawn from further consideration by the examiner, 37 CFR 1.142(b), as being drawn to a non-elected invention.

Claim Rejections - 35 USC 112

Claims 1-19 are rejected as failing to define the invention in the manner required by 35 U.S.C. 112. second paragraph.

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject mailer which the applicant regards as his invention.

The claim(s) are narrative in form and replete with indefinite and functional or operational language. The structure which goes to make up the device must be clearly and positively specified. The structure must be organized and correlated in such a manner as to present a complete operative device. The claim(s) must be in one sentence form only. Note the format of the claims in the patent(s) cited.

After carefully reviewing the listed items by very detailed work of the examiner, I have consulted with patent consultant and also tried my best to amend all claims with respect to those indefinite and other language issues in previous claims.

This way, the claims, as amended, therefore appears to overcome indefinite and other issues; withdrawal of the rejection of claims under 35 USC 112 is respectfully requested.

Regarding claim 16, it is unclear to the examiner what is being claimed. Applicant's preamble recites a method of packaging a guest OS image inside a host OS application, but the examiner fails to understand how the recited steps accomplish the recited goal. In particular, it is not at all clear how the recited steps of claim 16 provide a result of a guest OS image packages inside a host OS application. Additionally, applicant has recited "some special file formats" (see rejection of claim 1). In line 7, applicant uses exemplary language which does not properly limit the scope of the claims (such as record headers"). In line 10 of claim 16, applicant recites 'the wrapper host application". There is insufficient antecedent basis for this limitation in the claims.

Claims 16-19 have been amended to conform the requirement.

Claim Rejections - 35 USC 103

Claims 1-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Huang et al., U.S. Patent Application Publication No. 2002/0157001 Al, in view of Ohno et al., U.S. Patent No. 6,715,016 B1.

Regarding claim 1, Huang teaches a mobile device to run multiple operating systems, employing a method that comprises the steps of:

preparing a guest OS image, and packaging the image into a host OS file [Fig. 6, second operating system];

starting a boot loader for the host OS to read, unpack, load, and start the guest OS image [paragraph 0018];

exiting the guest OS, running exiting code to restore the system state and data to the original host OS [paragraph 0018].

Huang does not teach the saving of a state and data of the host OS before switching to the guest OS.

Ohno teaches a computer system for running multiple operating systems that saves the state of the current OS before switching to a second OS[col. 2, lines 4-11].

It would have been obvious to one of ordinary skill in the art to combine the teachings of Huang an Ohno by modifying Huang to save the state of the host OS before switching to the guest Os, as taught by Ohno. Both Huang and Ohno are directed towards analogous subject mailer — namely, running of multiple operating systems on a single computer, and enabling the switching between them. The teachings of Ohno would improve the system of Huang by providing a way to easily restore the operating state of the host OS when switching back from operating under the guest OS.

United States Patent Application 20020157001 by Huang, Alec ; et al. on October 24, 2002 titled "Computer system capable of switching operating system" talks about a computer system, having a portable computer and an expanding apparatus. The portable computer has a display, a first storage apparatus to store a first operating system and an operating system load program, a processing circuit to execute the first operating system for controlling operation of the computer system, and a connector. The expanding apparatus has a second storage apparatus to store a second operating system. When the connector of the portable computer is disconnected with the expanding apparatus, the processing circuit executes the first operating system to allow the portable computer operating independently. When the connector is coupled to the expanding apparatus, the processing circuit uses the operating system load program to load and execute the second operating system, while the first

operating system is terminated. In addition, a docking station may also be installed in the expanding apparatus to dispose the portable computer thereon.

Claim 1 has thus been amended to include additional steps as how to run another OS from within a host OS in mobile devices. Although Huang's method includes a portable computer and an expanding apparatus and its computer system includes a personal data assistant, the way it starts a second OS is more traditional and similar to the booting of multiple OS in desktop computers. Our invention has been targeted for the mobile devices and there are some significant differences from Huang's method:

First, in majority of mobile devices, memories are limited and traditional OS loading and booting techniques, which requires huge amount of memory usage, are inappropriate. Many mobile devices using flash memory as secondary storage for the device, normal OS booting strategy - loading OS images from secondary storage such as hard disk into system memory will only double the memories used.

Furthermore, as users using mobile devices normally expect instant responses, switching between OS shall be done as quickly as possible.

In lie of this, this invention prepare and package guest OS image so that guest OS image can execute in place; this way, code in guest OS can be executed directly from within flash memory without the need to reload programs into system RAM and the starting of guest OS can be faster.

While in Huang's method, when different Operating Systems stored in expanding apparatus are executed by the processing circuit, it just boot as a normal desktop OS and will terminate the previous OS from memory and thus consumes significant overhead. The booting and starting another OS in Huang's method has not much difference from dual boot system widely used in desktop computer systems today.

Second, many mobile devices use its own special application or data format and does not support reading or executing arbitrary files. For example,

Palm OS requires all files be packed into PRC or PDB format before it can be recognized. This way, simply storing an OS inside expanding apparatus, flash memory or hard disk cannot start the second OS from within the host OS itself.

So, in our invention, we package the guest OS into native application so as to preserve the execute-in-place characteristics of guest OS while allowing the application be recognizable by the host OS. This way, a user can start a guest OS as if it is a native application in the host OS.

Huang's method does not teach the use of native application to warp a guest OS so that the guest OS can start and execute in place from within the host OS. Instead, When the portable computer is connected to the expanding apparatus via the connector, the processing circuit displays a selection frame on the display for the user to decide which operating system is to be executed.

United States Patent 6,715,016 by Ohno, et al. on March 30, 2004 entitled "Multiple operating system control method" talks about *an inter-OS control software for switching OS's in operation executed on a single CPU is installed, and plural OS's are made alternately executed. A control program is executed exclusively on one OS, which controls the controlled apparatus. A supervisory control program and a development environment program are executed on another OS, and a memory space is divided so as to make no effect for the operation of the control program. A higher real-time performance and reliability can be established with a single CPU architecture.*

Ohno's methods teach storing and reloading context information of CPU operation. "*In responsive to the generation of interrupt or the request signal from OS's or the software programs running on OS's, this inter-OS control software stores and revises the context information of CPU operations (for example, register values of CPU), switches the memory spaces and restarts the OS operations in another context stored in past. In other words, the operation of the running OS is terminated and the operation of other OS's is restarted.*"

Ohno's method differs from our method in that it requires physical memory spaces be pre-defined and arranged before hand in a particular order.

"*Individual memory spaces occupied for the individual OS's and a memory space shared and accessible commonly by plural OS's are defined on the physical memory. Owing to this configuration, the individual memory spaces for the kernel and the programs are so defined that the interference among OS's such as data destruction may be avoided and that the necessary data may be shared by the programs each executed on the difference OS's.* "

"*In order to define the separated areas individually occupied by OS-A or OS-B, at first, OS-A is made started when the control apparatus 1 starts up, and then, the memory areas are so defined that the physical memory area lower than the designated address specified by the start-up option of OS-A may be occupied by OS-A. After start-up of OS-A, OS-B is made started by loading OS-B onto the physical memory area higher than the designated address. The kernel of OS-B acquires the designated address recorded on the table in the inter-OS control software 23, and uses the area which is not used by OS-B under the memory management.*"

Ohno's method has a limit to run targeted OS in many mobile devices because many mobile OS do not have capabilities to use only pre-define memories. In reality, the memory addresses used by an OS could be arbitrary addresses and may not be some continuous addresses already defined, or arranged.

This invention leverages the limit of defining memories used by both OS and allows much more flexibility for OS in the mobile device to execute another OS. Instead of defining what memory spaces to be used beforehand, this invention reserves memories already used by the host OS before executing the target OS. By this way, there is no need for both OS to define what memory spaces are to be used. From guest OS's viewpoint, reserved memory is either invisible or un-writable, there is no limit as what memory spaces must be in low area and what memory spaces must be in high area. Through this approach, host OS does not need to be modified to work with defined memories in order to

execute guest OS. Guest OS requires only minimum changes to recognize memory spaces already reserved. For mobile devices, this approach is more flexible because majority of mobile OS is stored in flash memory and is very hard to be modified to support defined memories. As Ohno's method uses static memory mappings for the whole system, this invention dynamically reserves and finds free memory spaces for the guest OS to use which is more suitable for mobile devices.

In view of these reasons, withdrawn of the rejection of claim 1 as being able to produce by combining the teachings of Huang and Ohno's methods appears in order and is respectfully requested.

Claims 2-15 have been rejected under 35 U.S.C. 103(a) for similar reasons and since they are dependent claims upon Claim 1. Now that Claim 1 have been amended and Claims 2-15 re-written, we respectfully request the withdrawn of rejection of Claims 2-15.

Conclusion

In view of the above reasons, it is submitted that claims 1-19 are allowable and applicant respectfully request an early notice to such effect. The applicant thanks very much for the work of the examiner, especially to much of the correction and recommendations on applicant's lack of patent practice or skills since the applicant is filing by individual inventor. The applicant would also greatly appreciate if the respectful examiner could tell if there is any points in the patent applications contain patentable or allowable points in the office action even there are further rejections on those amended claims.

Respectfully submitted,

Yongyong Xu

408-215-8485

yongyongxu@yahoo.com